Two simulations about DPLL(T)

Mahfuza Farooque, Stéphane Lengrand, Assia Mahboubi

Two simulations about DPLL($\mathcal{T}$)

Mahfuza Farooque[1], Stéphane Lengrand[1,2] and Assia Mahboubi[3]

[1] CNRS
[2] Ecole Polytechnique
[3] Microsoft Research - INRIA Joint Centre

Project PSI: "Proof Search control in Interaction with domain-specific methods"
ANR-09-JCJC-0006

23rd April 2012

Abstract

In this paper we relate different formulations of the DPLL($\mathcal{T}$) procedure.

The first formulation is that of [NOT06] based on a system of rewrite rules, which we denote DPLL($\mathcal{T}$).

The second formulation is an inference system of [Tin02], which we denote $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

The third formulation is the application of a standard proof-search mechanism in a sequent calculus $\mathsf{LK}^p(\mathcal{T})$ introduced here.

We formalise an encoding from DPLL($\mathcal{T}$) to $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ that was, to our knowledge, never explicitly given and, in the case where DPLL($\mathcal{T}$) is extended with backjumping and Lemma learning, never even implicitly given.

We also formalise an encoding from $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ to $\mathsf{LK}^p(\mathcal{T})$, building on Ivan Gazeau's previous work: we extend his work in that we handle the "-modulo-Theory" aspect of SAT-modulo-theory, by extending the sequent calculus to allow calls to a theory solver (seen as a blackbox). We also extend his work in that we handle advanced features of DPLL such as backjumping and Lemma learning, etc.

Finally, we refine the approach by starting to formalise quantitative aspects of the simulations: the complexity is preserved (nunber of steps to build complete proofs). Other aspects remain to be formalised (non-determinism of the search / width of search space).

Contents

1 Encoding DPLL($\mathcal{T}$) in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

In this section we encode $\mathsf{DPLL}(\mathcal{T})$ in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

Note that there exist different variants of $\mathsf{DPLL}(\mathcal{T})$. We first consider the basic version which is equipped with backtracking. This formalises ideas presented in [Tin02].

Then we enhance the encoding to the enhanced version of $\mathsf{DPLL}(\mathcal{T})$ with backjumping, a generalised version of backtracking.

The main gap between $\mathsf{DPLL}(\mathcal{T})$ and an inference system such as $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ is the fact that a (successful) $\mathsf{DPLL}(\mathcal{T})$ run is a rewrite sequence finishing with the state UNSAT, while a (successful) proof-search run is (/ produces) a proof tree. Roughly speaking, the $\mathsf{DPLL}(\mathcal{T})$ procedure implements the depth-first search of the corresponding tree.

1.1 Preliminaries: $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ and its properties

Definition 1 (The system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$) *Clauses* are finite disjunctions of literals considered up to commutativity and associativity. We will denote them C, C_0, C_1 etc; the empty clause will be denoted by $\bot$. The cardinality of a clause C is denoted $|C|$.

Finite sets of clauses, e.g. $\{C_1, \ldots, C_n\}$, will be denoted ϕ, ϕ_0, etc. By $|\phi|$ we denote the sum of the sizes of the clauses in ϕ. By $\mathsf{lit}(\phi)$ we denote the set of literals that appear in ϕ or whose negations appear in ϕ.

Given a theory $\mathcal{T}$ the system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$, given in Figure 1, is an inference system on sequents of the form $\Delta;\phi \vdash_{\mathcal{T}}$, where Δ is a set of literals (e.g. $\{l_1, \ldots, l_n\}$).

$$\frac{\Delta, l^{\bot};\phi \vdash_{\mathcal{T}} \qquad \Delta, l;\phi \vdash_{\mathcal{T}}}{\Delta;\phi \vdash_{\mathcal{T}}}\ \textit{Split} \text{ where } l \in \mathsf{lit}(\phi),\ \Delta, l^{\bot} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

$$\frac{}{\Delta;\phi, \bot \vdash_{\mathcal{T}}}\ \textit{Empty} \qquad \frac{\Delta, l;\phi, l \vdash_{\mathcal{T}}}{\Delta;\phi, l \vdash_{\mathcal{T}}}\ \textit{Assert} \text{ where } \Delta, l^{\bot} \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$$

$$\frac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}}\ \textit{Subsume} \text{ where } \Delta, l^{\bot} \models_{\mathcal{T}} \qquad \frac{\Delta;\phi, C \vdash_{\mathcal{T}}}{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}}\ \textit{Resolve} \text{ where } \Delta, l \models_{\mathcal{T}}$$

Figure 1: System $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

The *Assert* rule models the fact that every literal occurring as a unit clause in the current clause set must be satisfied for the whole clause set to be satisfied. The *Split* is mainly used to branch the proof tree from the DPLL rewrite sequence system.This rule corresponds to the decomposition in smaller subproblems of the DPLL method. This rule is the only $don't\ know\ non-deterministic$ rule of the calculus. The *Resolve* rule removes from a clause all literals whose complement has been asserted (which corresponds to generating the simplified clause by unit resolution and the discarding the clause by backword subsumption). The *Subsume* rule removes from the clauses that contain an asserted literal(because all of these clause will be satisfied in any model in which the asserted literal is true). To close the branch of a proof tree we use the *empty* rule is in the calculus just for convenience and could be removed with no loss of completeness. It models the fact that a derivation can be terminated as soon as the empty clause is derived. We do not consider that the model is consistent and satisfiable.

Definition 2 (Semantical entailment) $\Delta \models_{\mathcal{T}} C$ is a semantical notion of entailment for a particular theory $\mathcal{T}$, i.e. every $\mathcal{T}$-model of Δ is a $\mathcal{T}$-model of C. A theory lemma is a clause C such that $\emptyset \models_{\mathcal{T}} C$.

Lemma 1 (Weakening 1) *The following rule is size-preserving admissible in* $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

$$\frac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta;\phi, C \vdash_{\mathcal{T}}}$$

Proof: By induction on $\Delta;\phi \vdash_{\mathcal{T}}$. □

Definition 3 (Consequences) For every set Δ of literals l, let $\mathsf{Sat}(\Delta) = \{l | \Delta \models_{\mathcal{T}} l\}$ and $\mathsf{Sat}_\phi(\Delta) = \mathsf{Sat}(\Delta) \cap \mathsf{lit}(\phi)$.

Remark 2 If $\mathsf{Sat}(\Delta) = \mathsf{Sat}(\Delta')$ then $\Delta \models_{\mathcal{T}} l$ iff $\Delta' \models_{\mathcal{T}} l$

Lemma 3 (Weakening 2) *The following rule is size-preserving admissible in* $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$

$$\frac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta';\phi \vdash_{\mathcal{T}}} \; \mathsf{Sat}_\phi(\Delta) \subseteq \mathsf{Sat}_\phi(\Delta')$$

Proof: By induction on the derivation of $\Delta;\phi \vdash_{\mathcal{T}}$:

Resolve $\dfrac{\Delta;\phi, C \vdash_{\mathcal{T}}}{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}} \Delta, l \models_{\mathcal{T}}$

We assume $\mathsf{Sat}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$
from which we get $\mathsf{Sat}_{\phi,C}(\Delta) \subseteq \mathsf{Sat}_{\phi,C}(\Delta')$, so we can apply the induction hypothesis to construct

$$\frac{\Delta';\phi, C \vdash_{\mathcal{T}}}{\Delta';\phi, l \vee C \vdash_{\mathcal{T}}} \Delta', l \models_{\mathcal{T}}$$

The side-condition is a consequence of the assumption $\mathsf{Sat}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$.

Subsume $\dfrac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}} \Delta, l^\perp \models_{\mathcal{T}}$

We assume $\mathsf{Sat}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$
from which we get $\mathsf{Sat}_{\phi}(\Delta) \subseteq \mathsf{Sat}_{\phi}(\Delta')$, so we can apply the induction hypothesis to construct

$$\frac{\Delta';\phi \vdash_{\mathcal{T}}}{\Delta';\phi, l \vee C \vdash_{\mathcal{T}}} \Delta', l^\perp \models_{\mathcal{T}}$$

The side-condition is a consequence of the assumption $\mathsf{Sat}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$.

Assert $\dfrac{\Delta, l;\phi, l \vdash_{\mathcal{T}}}{\Delta;\phi, l \vdash_{\mathcal{T}}} \Delta, l^\perp \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$

We assume $\mathsf{Sat}_{\phi,l}(\Delta) \subseteq \mathsf{Sat}_{\phi,l}(\Delta')$
from which we get $\mathsf{Sat}_{\phi,l}(\Delta, l) \subseteq \mathsf{Sat}_{\phi,l}(\Delta', l)$.

- If $\Delta' \models_{\mathcal{T}} l$, then $\mathsf{Sat}(\Delta', l) = \mathsf{Sat}(\Delta')$, so we have $\mathsf{Sat}_{\phi,l}(\Delta, l) \subseteq \mathsf{Sat}_{\phi,l}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vdash_{\mathcal{T}}$.
- If $\Delta' \models_{\mathcal{T}} l^\perp$, then we construct

$$\dfrac{\dfrac{}{\Delta';\phi, \perp \vdash_{\mathcal{T}}}\, \mathit{Empty}}{\Delta';\phi, l \vdash_{\mathcal{T}}}\, \mathit{Resolve}$$

- If $\Delta' \not\models_{\mathcal{T}} l$ and $\Delta' \not\models_{\mathcal{T}} l^\perp$: we first apply the induction hypothesis to get $\Delta', l;\phi, l \vdash_{\mathcal{T}}$ and we conclude by constructing

$$\frac{\Delta', l;\phi, l \vdash_{\mathcal{T}}}{\Delta';\phi, l \vdash_{\mathcal{T}}} \Delta', l^\perp \not\models_{\mathcal{T}} \text{ and } \Delta', l \not\models_{\mathcal{T}}$$

Split $\dfrac{\Delta, l^\perp;\phi, l \vee C \vdash_{\mathcal{T}} \quad \Delta, l;\phi, l \vee C \vdash_{\mathcal{T}}}{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}} \Delta, l^\perp \not\models_{\mathcal{T}} \text{ and } \Delta, l \not\models_{\mathcal{T}}$

We assume $\mathsf{Sat}_{\phi,l\vee C}(\Delta) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$ from which we get both
$\mathsf{Sat}_{\phi,l\vee C}(\Delta, l) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta', l)$ and $\mathsf{Sat}_{\phi,l\vee C}(\Delta, l^\perp) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta', l^\perp)$.

- If $\Delta' \models_{\mathcal{T}} l$, then $\mathsf{Sat}(\Delta') = \mathsf{Sat}(\Delta', l)$, so we have $\mathsf{Sat}_{\phi,l\vee C}(\Delta, l) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vee C \vdash_{\mathcal{T}}$.
- If $\Delta' \models_{\mathcal{T}} l^\perp$, then $\mathsf{Sat}(\Delta') = \mathsf{Sat}(\Delta', l^\perp)$, so we have $\mathsf{Sat}_{\phi,l\vee C}(\Delta, l^\perp) \subseteq \mathsf{Sat}_{\phi,l\vee C}(\Delta')$. The induction hypothesis then gives $\Delta';\phi, l \vee C \vdash_{\mathcal{T}}$.

– If $\Delta' \not\models_{\mathcal{T}} l$ and $\Delta' \not\models_{\mathcal{T}} l^{\perp}$: the induction hypothesis on both premises gives $\Delta', l;\phi, l \vee C \vdash_{\mathcal{T}}$ and $\Delta', l^{\perp};\phi, l \vee C \vdash_{\mathcal{T}}$, and we can conclude

$$\frac{\Delta', l^{\perp};\phi, l \vee C \vdash_{\mathcal{T}} \quad \Delta', l;\phi, l \vee C \vdash_{\mathcal{T}}}{\Delta';\phi, l \vee C \vdash_{\mathcal{T}}} \Delta' \not\models_{\mathcal{T}} l \text{ and } \Delta' \not\models_{\mathcal{T}} l^{\perp}$$

Empty Straightforward.

□

Lemma 4 (Invertibility of Resolve) *Resolve is size-preserving invertible in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.*

Proof: By induction on the derivation of $\Delta;\phi, C \vee l \vdash_{\mathcal{T}}$ we prove $\Delta;\phi, C \vdash_{\mathcal{T}}$ (with the assumption $\Delta, l \models_{\mathcal{T}}$):

Resolve easily permutes with other instances of *Resolve* and with instances of *Subsume*.

Assert The side-condition of the rule guarantees that the literal added to the model, say l', is different from l:

$$\frac{\Delta, l';\phi', l', C \vee l \vdash_{\mathcal{T}}}{\Delta;\phi', l', C \vee l \vdash_{\mathcal{T}}} \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

We can construct

$$\frac{\Delta, l';\phi', l', C \vdash_{\mathcal{T}}}{\Delta;\phi', l', C \vdash_{\mathcal{T}}} \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

whose premiss is proved by the induction hypothesis.

Split $$\frac{\Delta, l'^{\perp};\phi, C \vee l \vdash_{\mathcal{T}} \quad \Delta, l';\phi, C \vee l \vdash_{\mathcal{T}}}{\Delta;\phi, C \vee l \vdash_{\mathcal{T}}} l' \in \mathsf{lit}(\phi, C \vee l) \text{ and } \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

We can construct

$$\frac{\Delta, l';\phi, C \vdash_{\mathcal{T}} \quad \Delta, l'^{\perp};\phi, C \vdash_{\mathcal{T}}}{\Delta;\phi, C \vdash_{\mathcal{T}}} l' \in \mathsf{lit}(\phi, C) \text{ and } \Delta, l'^{\perp} \not\models_{\mathcal{T}} \text{ and } \Delta, l' \not\models_{\mathcal{T}}$$

whose branches are closed by using the induction hypothesis. The side-condition $l' \in \mathsf{lit}(\phi, C)$ is satisfied because $l \neq l'$.

Empty Straightforward.

□

We now introduce a new system $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ which is an extended version of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ with $Weakening1$, $Weakening2$ and the *Inverted Resolve*. By the previous lemmas, a sequent derivable in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is derivable in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

$$\frac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta;\phi, C \vdash_{\mathcal{T}}} \qquad \frac{\Delta;\phi \vdash_{\mathcal{T}}}{\Delta';\phi \vdash_{\mathcal{T}}} \mathsf{Sat}_{\phi}(\Delta) \subseteq \mathsf{Sat}_{\phi}(\Delta') \qquad \frac{\Delta;\phi, l \vee C \vdash_{\mathcal{T}}}{\Delta;\phi, C \vdash_{\mathcal{T}}} \Delta, l \models_{\mathcal{T}}$$

Figure 2: System $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$

Definition 4 (Size of proof-trees in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$) The size of proof-trees in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is defined as the size of trees in the usual sense, but not counting the occurences of *Weakening1*, *Weakening2* or the *Inverted Resolve* rules.[1]

Remark 5 The size-preserving admissibility results of those three rules in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ entails that a proof-tree in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ of size n, can be transformed into a proof-tree in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ of size at most n.

Lemma 6 *If $\Delta \models_{\mathcal{T}} \neg C$ then there is a proof-tree concluding $\Delta;C, \phi \vdash_{\mathcal{T}}$ of size at most $|\phi| + 1$.*

[1] For that reason, dashed lines will be used for the occurences of those inference rules.

Proof: Here $\Delta \models_{\mathcal{T}} \neg C$ means $C = l_1 \vee \ldots \vee l_n$ and for all l_i, $\forall l_i\ \Delta \models_{\mathcal{T}} l_i^{\perp}$ where i=1,...,n. We can therefore construct

$$\dfrac{}{\Delta;\perp, \phi \vdash_{\mathcal{T}}}\ \text{Empty}$$
$$\dfrac{\Delta;\perp, \phi \vdash_{\mathcal{T}}}{\Delta;C, \phi \vdash_{\mathcal{T}}}\ \text{Resolve}$$

□

1.2 DPLL($\mathcal{T}$) with backtracking

In this section we describe the basic DPLL($\mathcal{T}$) procedure [NOT06], and its encoding into $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$.

Definition 5 (Basic DPLL($\mathcal{T}$)) *Models* are defined by the following grammar:

$$\Delta ::= \ () \mid \Delta, l^d \mid \Delta, l$$

where l ranges over literals, and l^d is an annotated literal called decision literal.

The basic DPLL($\mathcal{T}$) procedure rewrites *states* of the form $\Delta \| \phi$, with the following rewriting rules:

- Fail:
  $\Delta \| \phi, C \Rightarrow \mathsf{UNSAT}$, with $|\Delta| \models \neg C$ and there is no decision literal in Δ.
- Decide:
  $\Delta \| \phi \Rightarrow \Delta, l^d \| \phi$ where $l \notin \Delta$, $l^{\perp} \notin \Delta$, $l \notin \phi$ or $l^{\perp} \notin \phi$.
- Backtrack:
  $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l^{\perp} \| \phi, C$ if $|\Delta_1, l, \Delta_2| \models \neg C$ and no decision literal is in Δ_2.
- Unit propagation:
  $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ where $|\Delta| \models \neg C$, $l \notin \Delta$, $l^{\perp} \notin \Delta$.
- Theory Propagate:
  $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $|\Delta| \models_{\mathcal{T}} l$, $l \in \mathsf{lit}(\phi)$ and $l \notin \Delta, l^{\perp} \notin \Delta$.

where $|\Delta|$ denotes the result of erasing the annotations on decision literals, an operation defined in Figure 3.

$$\begin{array}{lcl} |()| & := & () \\ |\Delta, l| & := & |\Delta|, l \\ |\Delta, l^d| & := & |\Delta|, l \end{array}$$

Figure 3: Erasing annotations

We now proceed with the encoding of the basic DPLL($\mathcal{T}$) procedure as the construction of a derivation tree in System $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. The simulation could be be stated as follows:

If $\Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$ then there is a $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ proof of $|\Delta|; \phi \vdash_{\mathcal{T}}$ (i.e. there is no $\mathcal{T}$-model of ϕ extending Δ).

This is true; however, there is more information in $\Delta \| \phi \Rightarrow^* \mathsf{UNSAT}$ than in $|\Delta|; \phi \vdash_{\mathcal{T}}$, because the DPLL($\mathcal{T}$) sequence leading to UNSAT also backtracks on decision literals. This means that not only there is no $\mathcal{T}$- model of ϕ extending $|\Delta|$, but no matter how decision literals of Δ are changed, there is still no $\mathcal{T}$- model of ϕ that can be constructed. This notion is formalised by collecting the backtrack models as follows:

Definition 6 (Backtrack models) In Fig. 4 we define the interpretation of a model as a collection (formally, a multiset) of sets of literals.

Remark 7 We have $|\Delta| \in [\Delta]$ and $[\![\Delta]\!] \subseteq [\Delta]$.

We consider a notion of a partial proof-tree to step-by-step simulate DPLL($\mathcal{T}$) runs.

$$
\begin{array}{lcl}
[\![()]\!] & := & \emptyset \\
[\![\Delta, l]\!] & := & [\![\Delta]\!] \\
[\![\Delta, l^d]\!] & := & [\Delta, l^\perp] \\
 & & \\
[\Delta] & := & [\![\Delta]\!] \cup \{\!\{|\Delta|\}\!\}
\end{array}
$$

Figure 4: Collecting backtrack points

Definition 7 (Partial proof-tree) A partial proof-tree in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ is a tree labelled with sequents, whose leaves are tagged as either *open* or *closed*, and such that every node that is not an open leaf is an instance of the $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ rules.[2]

A complete proof-tree is a partial proof-tree whose leaves are all closed.

A partial proof-tree π' is an *n-extension* of π if π' is π or if π' is obtained from π by replacing one of its open leaves by a partial proof-tree of size at most n and whose conclusion has the same label as that leaf.

Definition 8 (Correspondence between DPLL($\mathcal{T}$) states and partial proof-trees) A partial proof-tree π *corresponds to* a $\mathsf{DPLL}(\mathcal{T})$ state $\Delta \| \phi$ if the sequents labelling its open leaves form a sub-set of $\{\Delta';\phi \vdash_{\mathcal{T}} \ \mid \Delta' \in [\Delta]\}$.

A partial proof-tree π corresponds to UNSAT if it has no open leaf.

The $\mathsf{DPLL}(\mathcal{T})$ procedure starts from an initial state i.e. $\emptyset \| \phi$, to which corresponds the partial proof-tree consisting of one node (both a root and a leaf) labelled with the sequent $;\phi \vdash_{\mathcal{T}}$.

Note that, different partial proof-trees might correspond to the same $\mathsf{DPLL}(\mathcal{T})$ state, as different $\mathsf{DPLL}(\mathcal{T})$ runs can lead to that state from various initial $\mathsf{DPLL}(\mathcal{T})$ states. The simulation theorem below expresses the fact that, when $\mathsf{DPLL}(\mathcal{T})$ rewrites one state to another state, any partial proof-tree corresponding to the formal state can be extended into a partial proof-tree corresponding to the latter state.

Theorem 8 *If $\Delta \| \phi \Rightarrow \mathcal{S}_2$ is a rewrite step of DPLL($\mathcal{T}$) and if π_1 corresponds to $\Delta \| \phi$ then there is, in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$, a $|\phi| + 1$-extension π_2 of π_1 corresponding to $\mathcal{S}_2$.*

Proof: By case analysis:

- Fail: $\Delta \| \phi, C \Rightarrow^* \mathsf{UNSAT}$ with $|\Delta| \models \neg C$ and there is no decision literal in Δ.

  Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi, C$. Since there are no decision literals in Δ, π_1 can have at most one open leaf, labelled by $|\Delta|;\phi, C \vdash_{\mathcal{T}}$.

  We $|\phi, C|$+1-extend π_1 into π_2 by replacing that leaf by a complete tree deriving $|\Delta|;\phi, C \vdash_{\mathcal{T}}$. We obtain that tree by applying Lemma 6 on the hypothesis $|\Delta| \models \neg C$. The new tree π_2 is complete and therefore corresponds to the UNSAT state of the $\mathsf{DPLL}(\mathcal{T})$ run.
- Decide: $\Delta \| \phi \Rightarrow \Delta, l^d \| \phi$ where $l \notin \Delta$, $l^\perp \notin \Delta$, $l \in \phi$ or $l^\perp \in \phi$.

  Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi$. We 1-extend it into π_2 by replacing the open leaf labelled with $|\Delta|;\phi \vdash_{\mathcal{T}}$ (if there is such a leaf) by one of three proof-trees:
  - If $|\Delta|, l \models_{\mathcal{T}}$, we have $\mathsf{Sat}(|\Delta|) = \mathsf{Sat}(|\Delta|, l^\perp)$ and we take:
    $$\dfrac{|\Delta|, l^\perp;\phi \vdash_{\mathcal{T}}}{|\Delta|;\phi \vdash_{\mathcal{T}}}\ \text{Weakening2}$$
    The new open leaves form a sub-set of $\{|\Delta|, l^\perp;\phi \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \ \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \ \mid \Delta' \in [\Delta, l^d]\}$ (since $|\Delta|, l^\perp = |\Delta, l^\perp| \in [\Delta, l^\perp] = [\![\Delta, l^d]\!] \subseteq [\Delta, l^d]$) and therefore π_2 corresponds to $\Delta, l^d \| \phi$.
  - If $|\Delta|, l^\perp \models_{\mathcal{T}}$, we have $\mathsf{Sat}(|\Delta|) = \mathsf{Sat}(|\Delta|, l)$ and we take
    $$\dfrac{|\Delta|, l;\phi \vdash_{\mathcal{T}}}{|\Delta|;\phi \vdash_{\mathcal{T}}}\ \text{Weakening2}$$

[2] A partial proof-tree that has no open leaf is isomorphic to a derivation in $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$.

The new open leaves form a sub-iset of $\{|\Delta|, l;\phi \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta, l^d]\}$ (since $|\Delta|, l = |\Delta, l| \in [\Delta, l^d]$) and therefore π_2 corresponds to $\Delta, l^d \| \phi$.

– If $|\Delta|, l \not\models_{\mathcal{T}}$ and $|\Delta|, l^\perp \not\models_{\mathcal{T}}$, we take

$$\frac{|\Delta|, l;\phi \vdash_{\mathcal{T}} \quad |\Delta|, l^\perp;\phi \vdash_{\mathcal{T}}}{|\Delta|;\phi \vdash_{\mathcal{T}}} \text{ Split}$$

The new open leaves form a sub-set of $\{|\Delta|, l;\phi \vdash_{\mathcal{T}} \} \cup \{|\Delta|, l^\perp;\phi \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta, l^d]\}$ and therefore π_2 corresponds to $\Delta, l^d \| \phi$. (since $|\Delta|, l^\perp = |\Delta, l^\perp| \in [\Delta, l^\perp] = [\![\Delta, l^d]\!] \subseteq [\Delta, l^d]$)

- Backtrack: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l^\perp \| \phi, C$ if $|\Delta_1, l, \Delta_2| \models \neg C$ and no decision literal is in Δ_2.

Let π_1 be a partial proof-tree corresponding to $\Delta_1, l^d, \Delta_2 \| \phi, C$. Since there are no decision literal in Δ_2, π_1 can have at most one open leaf, labelled with $|\Delta_1, l^d, \Delta_2|;\phi, C \vdash_{\mathcal{T}}$. We $|\phi, C|$+1-extend π_1 into π_2 by replacing that leaf by a complete tree deriving $|\Delta_1, l^d, \Delta_2|;\phi, C \vdash_{\mathcal{T}}$. We obtain that partial proof-tree by applying lemma 6 on the assumption $|\Delta_1, l^d, \Delta_2| \models \neg C$.

The new open leaves form a sub-set of $\{|\Delta|_1, l^\perp;\phi, C \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta_1, l^d, \Delta_2]\!]\} = \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta_1, l^d]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta_1, l^\perp]\}$ (since $|\Delta_1|, l^\perp = |\Delta_1, l^\perp| \in [\Delta_1, l^\perp]$) and therefore π_2 corresponds to $\Delta_1, l^\perp \| \phi, C$ state of the $\mathsf{DPLL}(\mathcal{T})$ run.

- Unit propagation : $\Delta \| \phi, C \vee l \Rightarrow \Delta, l \| \phi, C \vee l$ where $|\Delta| \models \neg C$, $l \notin \Delta$, $l^\perp \notin \Delta$.

Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi, C \vee l$. We $|\phi, C \vee l|$+1-extend it into π_2 by replacing the open leaf labelled with $|\Delta|;\phi, C \vee l \vdash_{\mathcal{T}}$ (if there is such a leaf) by one of three proof-trees:

– If $|\Delta|, l^\perp \models_{\mathcal{T}}$, we have $\mathsf{Sat}(|\Delta|) = \mathsf{Sat}(|\Delta|, l)$ and we take:

$$\frac{|\Delta|, l;\phi \vdash_{\mathcal{T}}}{|\Delta|;\phi \vdash_{\mathcal{T}}} \text{ Weakening2}$$

The new open leaves form a sub-set of $\{|\Delta|, l;\phi, C \vee l \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi, C \vee l \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi, C \vee l \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta, l]\}$ (since $|\Delta|, l = |\Delta, l| \in [\Delta, l]$) and therefore π_2 corresponds to $\Delta, l \| \phi, C \vee l$.

– If $|\Delta|, l \models_{\mathcal{T}}$ then lemma 6 directly provides a partial proof-tree of $|\Delta|;\phi, C \vee l \vdash_{\mathcal{T}}$.

– If $|\Delta|, l \not\models_{\mathcal{T}}$ and $|\Delta|, l^\perp \not\models_{\mathcal{T}}$, we can construct the following tree:

$$\cfrac{\cfrac{\cfrac{|\Delta|, l;\phi, C \vee l \vdash_{\mathcal{T}}}{|\Delta|, l;\phi, l \vdash_{\mathcal{T}}} \text{ Inverted Resolve}}{|\Delta|;\phi, l \vdash_{\mathcal{T}}} \text{ Assert}}{|\Delta|;\phi, C \vee l \vdash_{\mathcal{T}}} \text{ Resolve}$$

where the side-conditions of *Resolve* are provided by the hypothesis $\Delta'' \models \neg C$.

The new open leaves form a sub-set of $\{|\Delta|, l;\phi, C \vee l \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta, l]\}$ and therefore π_2 corresponds to $\Delta, l \| \phi, C \vee l$. (since $|\Delta|, l = |\Delta, l| \in [\Delta, l]$)

- Theory Propagate: $\Delta \| \phi \Rightarrow \Delta, l \| \phi$ where $|\Delta| \models_{\mathcal{T}} l$, $l \in \mathsf{lit}(\phi)$ and $l \notin \Delta, l^\perp \notin \Delta$.

Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi$. We 1-extend it into π_2 by replacing the open leaf labelled with $|\Delta|;\phi \vdash_{\mathcal{T}}$ by the following proof-tree :

$$\frac{|\Delta|, l;\phi \vdash_{\mathcal{T}}}{|\Delta|;\phi \vdash_{\mathcal{T}}} \text{ Weakening2}$$

The new open leaves form a sub-set of $\{|\Delta|, l;\phi \vdash_{\mathcal{T}} \} \cup \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta]\!]\} \subseteq \{\Delta';\phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta, l]\}$ (since $|\Delta|, l = |\Delta, l| \subseteq [\Delta, l]$) and therefore π_2 corresponds to $\Delta, l \| \phi$.

□

Corollary 9 *$LK_{DPLL}(\mathcal{T})$ is complete, i.e. if $\phi \models_{\mathcal{T}}$ then ;$\phi \vdash_{\mathcal{T}}$.*

Proof: By completeness of basic $\mathsf{DPLL}(\mathcal{T})$ and Theorem 8. □

1.3 $\mathsf{DPLL}(\mathcal{T})$ with backjumping and Lemma learning

We now consider a more advanced version of $\mathsf{DPLL}(\mathcal{T})$, which involves backjumping and lemma learning features, and which we denote $\mathsf{DPLL}_{bj}(\mathcal{T})$. $\mathsf{DPLL}_{bj}(\mathcal{T})$ extends basic $\mathsf{DPLL}(\mathcal{T})$ with the rules known as $\mathcal{T}$-Backjump, $\mathcal{T}$-Learn, $\mathcal{T}$-Forget, and Restart [NOT06]. Those rules drastically increase the efficiency of SMT-solvers.

$\mathcal{T}$-Backjump: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l_{bj} \| \phi, C$ with

1. $|\Delta_1, l^d, \Delta_2| \models \neg C$.
2. $|\Delta_1| \models \neg C'$
3. $\phi, C \models_{\mathcal{T}} C' \vee l_{bj}$
4. $l_{bj} \notin \Delta_1$, $l_{bj}^{\perp} \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.

for some clause C' such that $\mathsf{lit}(C') \subset \mathsf{lit}(\phi, C)$.

$\mathcal{T}$-Learn: $\Delta \| \phi \Rightarrow \Delta \| \phi, C$ if $\mathsf{lit}(C) \subseteq \mathsf{lit}(\phi, \Delta)$ and $\phi \models_{\mathcal{T}} C$.

$\mathcal{T}$-Forget: $\Delta \| \phi, C \Rightarrow \Delta \| \phi$ if $\phi \models_{\mathcal{T}} C$.

Restart: $\Delta \| \phi \Rightarrow \emptyset \| \phi$.

In order to simulate those extra rules in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$, we need to extend $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ with a cut rule as follows:

Definition 9 ($\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ with cut) System $\mathsf{LK}^{c}_{\mathsf{DPLL}}(\mathcal{T})$ is obtained by extending system $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ with the following cut-rule:

$$\frac{\Delta;\phi, l_1, \ldots, l_n \vdash_{\mathcal{T}} \quad \Delta;\phi, C \vdash_{\mathcal{T}}}{\Delta;\phi \vdash_{\mathcal{T}}} \mathit{Cut} \text{ where } C = l_1^{\perp}, \ldots, l_n^{\perp}$$

We define the size of proof-trees in $\mathsf{LK}^{c}_{\mathsf{DPLL}}(\mathcal{T})$ as we did for $\mathsf{LK}_{\mathsf{DPLL}^+}(\mathcal{T})$ (ignoring *Weakening1*, *Weakening2* or the *Inverted Resolve*), but also ignoring the left-branch of the cut-rules.[3]

Definition 10 ($n, \phi, \mathcal{S}$-sync action) π_ϕ is a $n, \phi, \mathcal{S}$-sync action if it is a function that maps every model $\Delta \in \mathcal{S}$ to a partial proof-tree of size at most n and concluding $\Delta;\phi \vdash_{\mathcal{T}}$.

Definition 11 (Parallel n-extension of partial proof-trees) π_2 is a *parallel n-extension* of π_1 according to π_ϕ if π_ϕ is a $n, \phi, \mathcal{S}$-sync action and if π_2 is obtained from π_1 by replacing all the open leaves of π_1 labelled by sequents of the form $\Delta;\phi \vdash_{\mathcal{T}}$ (where $\Delta \in \mathcal{S}$) by $\pi_\phi(\Delta)$.

Theorem 10 *If $\Delta \| \phi \Rightarrow_{DPLL_{bj}(\mathcal{T})} \mathcal{S}_2$ and π_1 corresponds to $\Delta \| \phi$, there is parallel $|\phi| + 3$-extension π_2 of π_1 (according to some π_ϕ) such that π_2 corresponds to $\mathcal{S}_2$.*

Proof: Since $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ is a sub-system of $\mathsf{LK}^{c}_{\mathsf{DPLL}}(\mathcal{T})$, we only need to simulate (in $\mathsf{LK}^{c}_{\mathsf{DPLL}}(\mathcal{T})$) the new rules.

$\mathcal{T}$-Backjump: $\Delta_1, l^d, \Delta_2 \| \phi, C \Rightarrow \Delta_1, l_{bj} \| \phi, C$ with

1. $|\Delta_1, l^d, \Delta_2| \models \neg C$.
2. $|\Delta_1| \models \neg C'$
3. $\phi, C \models_{\mathcal{T}} C' \vee l_{bj}$
4. $l_{bj} \notin \Delta_1$, $l_{bj}^{\perp} \notin \Delta_1$ and $l_{bj} \in \mathsf{lit}(\phi, \Delta_1, l^d, \Delta_2)$.

[3] As we shall see in the simulation theorem, this definition mimicks the fact that the length of $\mathsf{DPLL}(\mathcal{T})$ sequences is a complexity measure that ignores the cost of checking the side-conditions.

Let π_1 be a partial proof-tree corresponding to $\Delta_1, l^d, \Delta_2 \| \phi, C$. We have to build a π_2 that corresponds to $\Delta_1, l_{bj} \| \phi, C$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta'; \phi, C \vdash_{\mathcal{T}}$ where $\Delta' \in [\Delta_1, l_{bj}]$.
Let $\mathcal{S} = [\Delta_1, l^d, \Delta_2] \backslash [\![\Delta_1]\!]$ and π_ϕ be the $|\phi, C|+3$, ϕ, C,$\mathcal{S}$-sync action that maps every $\Delta \in \mathcal{S}$ to

$$\dfrac{\dfrac{;\phi, C, \neg C', l_{bj}^{\perp} \vdash_{\mathcal{T}}}{|\Delta_1|; \phi, C, \neg C', l_{bj}^{\perp} \vdash_{\mathcal{T}}} Weakening2 \qquad \dfrac{\dfrac{\dfrac{|\Delta_1|, l_{bj}; \phi, C \vdash_{\mathcal{T}}}{|\Delta_1|, l_{bj}; \phi, C, l_{bj} \vdash_{\mathcal{T}}} Subsume}{|\Delta_1|; \phi, C, l_{bj} \vdash_{\mathcal{T}}} Assert}{|\Delta_1|; \phi, C, C' \vee l_{bj} \vdash_{\mathcal{T}}} Resolve}{|\Delta_1|; \phi, C \vdash_{\mathcal{T}}} cut}{\Delta; \phi, C \vdash_{\mathcal{T}}} Weakening2$$

It is a valid partial proof-tree because $\Delta \in \mathcal{S}$ entails $|\Delta_1| \subseteq \Delta$ and therefore $\mathsf{Sat}_\phi(|\Delta_1|) \subseteq \mathsf{Sat}_\phi(\Delta)$. The left branch is closed by assumption (3) and the completeness of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ on $\phi, C, \neg C', l_{bj}^{\perp} \models_{\mathcal{T}}$ (Corollary 9). We cannot anticipate the size of the proof-tree closing that branch, and we therefore ignore that proof-tree to compute the size of the whole tree, just as the length of the $\mathsf{DPLL}(\mathcal{T})$ run ignores the cost of checking $\phi, C \models_{\mathcal{T}} C' \vee l_{bj}$.

Let π_2 be the *parallel* $|\phi, C|+3$-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{|\Delta_1|, l_{bj}; \phi, C \vdash_{\mathcal{T}} \} \cup \{\Delta'; \phi \vdash_{\mathcal{T}} \mid \Delta' \in [\![\Delta_1]\!]\} \subseteq \{\Delta'; \phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta_1, l_{bj}]\}$ (since $|\Delta_1|, l_{bj} = |\Delta_1, l_{bj}| \in [\Delta_1, l_{bj}]$ and $[\![\Delta_1, l_{bj}]\!] = [\![\Delta_1]\!]$) and therefore π_2 corresponds to $\Delta_1, l_{bj} \| \phi, C$.

$\mathcal{T}$-Learn: $\Delta \| \phi \Rightarrow \Delta \| \phi, C$ if each atom of C occurs in ϕ or in Δ and $\phi \models_{\mathcal{T}} C$.
Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi$. We have to build a π_2 that corresponds to $\Delta \| \phi, C$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta'; \phi, C \vdash_{\mathcal{T}}$ where $\Delta' \in [\Delta]$.
Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $|\phi|$,ϕ,$\mathcal{S}$-sync actionthat maps every $\Delta \in \mathcal{S}$ to:

$$\dfrac{\dfrac{;\phi, \neg C \vdash_{\mathcal{T}}}{\Delta; \phi, \neg C \vdash_{\mathcal{T}}} Weakening2 \qquad |\Delta|; \phi, C \vdash_{\mathcal{T}}}{\Delta; \phi \vdash_{\mathcal{T}}} cut$$

The left branch of the cut is closed by assumption and completeness of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ on $\phi, \neg C \models_{\mathcal{T}}$ (Corollary 9). We cannot anticipate the size of the proof-tree closing that branch, and we therefore ignore that proof-tree to compute the size of the whole tree, just as the length of the $\mathsf{DPLL}(\mathcal{T})$ run ignores the cost of checking $\phi \models_{\mathcal{T}} C$.

Let π_2 be the *parallel* $|\phi|$-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{\Delta'; \phi, C \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta]\}$ and therefore π_2 corresponds to $\Delta \| \phi, C$. .

$\mathcal{T}$-Forget: $\Delta \| \phi, C \Rightarrow \Delta \| \phi$ if $\phi \models_{\mathcal{T}} C$.

Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi, C$. We have to build a π_2 that corresponds to $\Delta \| \phi$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $\Delta'; \phi \vdash_{\mathcal{T}}$ where $\Delta' \in [\Delta]$.
Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $1, \phi, C$,$\mathcal{S}$-sync action that maps every $\Delta' \in \mathcal{S}$ to

$$\dfrac{\Delta'; \phi \vdash_{\mathcal{T}}}{\Delta'; \phi, C \vdash_{\mathcal{T}}} Weakening1$$

Let π_2 be the *parallel* 1-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{\Delta'; \phi \vdash_{\mathcal{T}} \mid \Delta' \in [\Delta]\}$ and therefore π_2 corresponds to $\Delta \| \phi$.

Restart: $\Delta \| \phi \Rightarrow \emptyset \| \phi$.

Let π_1 be a partial proof-tree corresponding to $\Delta \| \phi$. We have to build a π_2 that corresponds to $\emptyset \| \phi$ in the $\mathsf{DPLL}_{bj}(\mathcal{T})$ run. This means that the open leaves of π_2 should be labelled with sequents of the form $;\phi \vdash_{\mathcal{T}}$

Let $\mathcal{S} = [\Delta]$ and π_ϕ be the $1, \phi$,$\mathcal{S}$-sync action that maps every $\Delta' \in \mathcal{S}$ to:

$$\frac{;\phi \vdash_{\mathcal{T}}}{\Delta';\phi \vdash_{\mathcal{T}}} - Weakening2$$

Let π_2 be the *parallel* 1-extension of π_1 according to π_ϕ. The new open leaves form a sub-set of $\{;\phi \vdash_{\mathcal{T}}\}$ and therefore π_2 corresponds to $\emptyset \| \phi$.

□

2 Encoding $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$

2.1 Preliminaries: System $\mathsf{LK}^p(\mathcal{T})$

In this section we introduce (the propositional fragment of) system $\mathsf{LK}^p(\mathcal{T})$.

Definition 12 (Formulae, negation) The formulae of $\mathsf{LK}^p(\mathcal{T})$ are given by the following grammar:

$$\text{Formulae} \quad A, B, \ldots \quad ::= \quad l \mid A \wedge^+ B \mid A \vee^+ B \mid A \wedge^- B \mid A \vee^- B$$

where l ranges over literals.

Let $\mathcal{P}$ be a set of literals declared to be *positive*, while their negations, required to not be in $\mathcal{P}$, are declared to be *negative*. Given such a set $\mathcal{P}$, we define positive formulae and negative formulae as the formulae generated by the following grammars:

$$\begin{array}{lll} \text{positive formulae} & P, \ldots & ::= \quad p \mid A \wedge^+ B \mid A \vee^+ B \\ \text{negative formulae} & N, \ldots & ::= \quad p^\perp \mid A \wedge^- B \mid A \vee^- B \end{array}$$

where p ranges over $\mathcal{P}$.

Negation is recursively extended into a involutive map from formulae to formulae as follows:

$(A \wedge^+ B)^\perp$	$:= \; A^\perp \vee^- B^\perp$	$(A \wedge^- B)^\perp$	$:= \; A^\perp \vee^+ B^\perp$
$(A \vee^+ B)^\perp$	$:= \; A^\perp \wedge^- B^\perp$	$(A \vee^- B)^\perp$	$:= \; A^\perp \wedge^+ B^\perp$

Definition 13 (System $\mathsf{LK}^p(\mathcal{T})$) The sequent calculus $\mathsf{LK}^p(\mathcal{T})$ has two kinds of sequents:

$\Gamma \vdash_{\mathcal{T}} [P]$ where P is in the *focus* of the sequent

$\Gamma \vdash_{\mathcal{T}} \Gamma'$

Its rules are given in Figure 5.

$\mathcal{T}(\Delta)$ is the call to the decision procedure on the conjunction of all atomic formulae within Δ. It holds if the procedure returns UNSAT.

$$\frac{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [A] \qquad \Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [B]}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [A \wedge^+ B]} \qquad \frac{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [A_i]}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [A_1 \vee^+ A_2]}$$

$$\frac{}{\Gamma, p \vdash^{\mathcal{P},p}_{\mathcal{T}} [p]} \qquad \frac{\mathcal{T}(\Gamma, p^\perp)}{\Gamma \vdash^{\mathcal{P},p}_{\mathcal{T}} [p]}$$

$$\frac{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} N}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} [N]} \; N \text{ negative}$$

$$\frac{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} A, \Delta \qquad \Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} B, \Delta}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} A \wedge^- B, \Delta} \qquad \frac{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} A_1, A_2, \Delta}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} A_1 \vee^- A_2, \Delta} \qquad \frac{\Gamma, A^\perp \vdash^{\mathcal{P}}_{\mathcal{T}} \Delta}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}} A, \Delta} \; A \text{ positive or atom}$$

$$\frac{\Gamma \vdash^{\mathcal{P},l}_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}}} \qquad \frac{\Gamma, P^\perp \vdash^{\mathcal{P}}_{\mathcal{T}} [P]}{\Gamma, P^\perp \vdash^{\mathcal{P}}_{\mathcal{T}}} \; P \text{ positive} \qquad \frac{\mathcal{T}(\Gamma)}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}}}$$

Figure 5: System $\mathsf{LK}^p(\mathcal{T})$

We also consider two cut-rules. The analytic cut:

$$\frac{\Gamma, l \vdash^{\mathcal{P}}_{\mathcal{T}} \quad \Gamma, l^\perp \vdash^{\mathcal{P}}_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}}}$$

with the condition that l appears in Γ.

The general cut:

$$\frac{\Gamma, l_1, \ldots, l_n \vdash^{\mathcal{P}}_{\mathcal{T}} \quad \Gamma, ({l_1}^{\perp} \vee^{-} \ldots \vee^{-} {l_n}^{\perp}) \vdash^{\mathcal{P}}_{\mathcal{T}}}{\Gamma \vdash^{\mathcal{P}}_{\mathcal{T}}}$$

2.2 Simulation

We now encode $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ in $\mathsf{LK}^p(\mathcal{T})$.

The main gap between $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ (or even $\mathsf{DPLL}(\mathcal{T})$) and a sequent calculus such as $\mathsf{LK}^p(\mathcal{T})$ is the fact that the structures handled by the former are very flexible (e.g. clauses are multisets of literals), while sequent calculus implements a root-first decomposition of formulae trees.

Clauses in $\mathsf{DPLL}(\mathcal{T})$ (and in $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$) are disjunctions considered modulo associativity and commutativity. The way we encode them as formulae of sequent calculus is as follows: a clause C will be represented by a formula C' which is a disjunctive tree whose leaves contain at least all the literals of C but also other literals that we can consider as garbage.

Of course, one could fear that the presence of garbage parts within C' degrades the efficiency of proof-search when simulating $\mathsf{DPLL}(\mathcal{T})$. This garbage comes from the original clauses at the start of the $\mathsf{DPLL}(\mathcal{T})$ rewriting sequence, which might have been simplified in later steps of $\mathsf{DPLL}(\mathcal{T})$ but which remain unchanged in sequent calculus. The size of the garbage is therefore smaller than the size of the original problem. We ensure that the inspection, by the proof-search process, of the garbage in ${C'}^{\perp}$, takes no more inference steps than the size of the garbage itself (the waste of time is linear in the size of the garbage). In order to ensure this, we use polarities and the focusing properties of $\mathsf{LK}^p(\mathcal{T})$: the garbage literals in C' must be negative atoms that are negated in the model/context.

Definition 14 ($\mathcal{P}$-correspondence) Let $\mathcal{P}$ be a multiset of literals.

- A formula C' $\mathcal{P}$-corresponds to a clause C (in system $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$), where $C = l_1 \vee \ldots \vee l_p$, if $C' = l'_1 \vee^{-} \ldots \vee^{-} l'_{p'}$ with $\{l_j\}_{j=1\ldots p} \subseteq \{l'_j\}_{j=1\ldots p'}$ and for any $l \in \{l'_j\}_{j=1\ldots p'} \backslash \{l_j\}_{j=1\ldots p}$, $l^{\perp} \in \mathcal{P}$.
- A $\mathsf{LK}^p(\mathcal{T})$ sequent $\Delta, {C'}_1, \ldots, {C'}_m \vdash^{\mathcal{P}}_{\mathcal{T}}$ corresponds to a $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$ sequent $\Delta; C_1, \ldots, C_m \vdash_{\mathcal{T}}$, if C'_i $\mathcal{P}$-corresponds to C_i and for all $l \in \mathcal{P}$, $\Delta \models_{\mathcal{T}} l$.

Lemma 11 *If C' $\mathcal{P}$-corresponds to C, then C' also $(\mathcal{P}, l)$-corresponds to C.*

Proof: Straightforward. □

Theorem 12 *Assume $\dfrac{\mathcal{S}_i}{\mathcal{S}}$ is a rule of $\mathsf{LK}_{\mathsf{DPLL}}(\mathcal{T})$. For every $\mathsf{LK}^p(\mathcal{T})$ sequent $\mathcal{S}'$ that corresponds to $\mathcal{S}$, there exist a partial proof-tree in $\mathsf{LK}^p(\mathcal{T})$*

- *whose open leaves $(\mathcal{S}'_i)$ are such that $\forall i$, $\mathcal{S}'_i$ corresponds to $\mathcal{S}_i$ and*
- *whose size is smaller than size $(\mathcal{S}') + 4$.*

Proof: By case analysis:

- Split:

  $$\frac{\Delta, l^{\perp}; \phi \vdash_{\mathcal{T}} \quad \Delta, l; \phi \vdash_{\mathcal{T}}}{\Delta; \phi \vdash_{\mathcal{T}}} \text{ where } l \in \mathsf{lit}(\phi), \Delta, l^{\perp} \nvdash_{\mathcal{T}} \text{ and } \Delta, l \nvdash_{\mathcal{T}}$$

  Assume that $\Delta, \phi' \vdash^{\mathcal{P}}_{\mathcal{T}}$ corresponds to $\Delta; \phi \vdash_{\mathcal{T}}$ (i.e. $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$).
  We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation that uses an analytic cut:

  $$\frac{\Delta, l^{\perp}, \phi' \vdash^{\Delta_0}_{\mathcal{T}} \quad \Delta, l, \phi' \vdash^{\Delta_0}_{\mathcal{T}}}{\Delta, \phi' \vdash^{\Delta_0}_{\mathcal{T}}}$$

  and $\Delta, l^{\perp}, \phi' \vdash^{\Delta_0}_{\mathcal{T}}$ $\mathcal{P}$-corresponds to $\Delta, l^{\perp}; \phi \vdash_{\mathcal{T}}$ and $\Delta, l, \phi' \vdash^{\Delta,_0}_{\mathcal{T}}$ $\mathcal{P}$-corresponds to $\Delta, l; \phi \vdash_{\mathcal{T}}$.

- Assert:

$$\frac{\Delta, l; \phi, l \vdash_{\mathcal{T}}}{\Delta; \phi, l \vdash_{\mathcal{T}}} \Delta, l^{\perp} \nvdash_{\mathcal{T}} \text{ and } \Delta, l \nvdash_{\mathcal{T}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vdash_{\mathcal{T}}$. (i.e. $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$, and C' $\mathcal{P}$-corresponds to l, that is to say $C' = \vee_{i=1}^{p} l_i$ where $l = l_{i_0}$ for some $i_0 \in 1 \ldots n$)

We build in LK^p the following derivation:

$$\dfrac{\dfrac{\dfrac{\mathcal{T}(\Delta, \phi', C', l_i)}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}} [l_i^{\perp}]}\, i \neq i_0 \qquad \dfrac{\dfrac{l_{i_0}, \Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}}}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}} l_{i_0}{}^{\perp}}}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}} [l_{i_0}{}^{\perp}]}}{\vdots \wedge^{+}.}}{\dfrac{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}} [C'^{\perp}]}{\dfrac{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}}}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}}}}}$$

For $i \neq i_0$, $l_i^{\perp} \in \Delta_0$, so it is positive and we can use an axiom (remember that $\Delta \models l_i^{\perp}$).

- Empty$_{\mathcal{T}}$:

$$\frac{}{\Delta; \phi, \perp \vdash_{\mathcal{T}}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, \perp \vdash_{\mathcal{T}}$ (i.e. C' $\mathcal{P}$-corresponds to $\perp$, $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i, $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$).

We build in LK^p the following derivation:

$$\dfrac{\dfrac{\dfrac{\mathcal{T}(\Delta, \phi', C', l_i)}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\Delta_0} [l_i^{\perp}]}}{\vdots \wedge^{+}.}}{\dfrac{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l_{i_0}} [C'^{\perp}]}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}}}}$$

Again, $l_i^{\perp} \in \Delta_0$, so it is positive and we can use an axiom (remember that $\Delta \models l_i^{\perp}$).

- Resolve:

$$\frac{\Delta; \phi, C \vdash_{\mathcal{T}}}{\Delta; \phi, l \vee C \vdash_{\mathcal{T}}} \Delta, l \models_{\mathcal{T}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vee C \vdash_{\mathcal{T}}$ (i.e. C' $\mathcal{P}$-corresponds to $l \vee C$, $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$). We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation

$$\frac{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}, l^{\perp}}}{\Delta, \phi', C' \vdash_{\mathcal{T}}^{\mathcal{P}}} \mathit{pol}$$

It suffices to notice that $\Delta, \phi', C' \vdash^{\mathcal{P}, l^{\perp}}$ corresponds to $\Delta; \phi, C \vdash_{\mathcal{T}}$.

- Subsume:

$$\frac{\Delta; \phi \vdash_{\mathcal{T}}}{\Delta; \phi, l \vee C \vdash_{\mathcal{T}}} \Delta, l^{\perp} \models_{\mathcal{T}}$$

Assume that $\Delta, \phi', C' \vdash^{\mathcal{P}}$ corresponds to $\Delta; \phi, l \vee C \vdash_{\mathcal{T}}$ (i.e. C' $\mathcal{P}$-corresponds to $l \vee C$, $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$).

- Cut: If we want to simulate $\mathsf{DPLL}(\mathcal{T})$ with backjump, we need to encode the cut rule of $\mathsf{LK}^c_{\mathsf{DPLL}}$.

$$\frac{\Delta;\phi, l_1, \ldots, l_n \vdash_{\mathcal{T}} \quad \Delta;\phi, C \vdash_{\mathcal{T}}}{\Delta;\phi \vdash} C = l_1^{\perp} \vee \ldots \vee l_n^{\perp}$$

Assume that $\Delta, \phi' \vdash_{\mathcal{T}}^{\mathcal{P}}$ corresponds to $\Delta;\phi \vdash_{\mathcal{T}}$ (i.e. $\phi' = C'_1, \ldots, C'_n$ and $\phi = C_1, \ldots, C_n$ with C'_i $\mathcal{P}$-corresponding to C_i for $i = 1 \ldots n$).

We build in $\mathsf{LK}^p(\mathcal{T})$ the following derivation that uses a general cut:

$$\frac{\Delta, \phi', l_1, \ldots, l_n \vdash^{\mathcal{P}} \quad \Delta, \phi', ({l_1}^{\perp} \vee^{-} \ldots \vee^{-} {l_n}^{\perp}) \vdash^{\mathcal{P}}}{\Delta, \phi' \vdash^{\mathcal{P}}} \mathit{cut}$$

Clearly, $\Delta, \phi', l_1, \ldots, l_n \vdash_{\mathcal{T}}^{\mathcal{P}}$ corresponds to $\Delta;\phi, l_1, \ldots, l_n \vdash_{\mathcal{T}}$ and $\Delta, \phi', ({l_1}^{\perp} \vee^{-} \ldots \vee^{-} {l_n}^{\perp}) \vdash_{\mathcal{T}}^{\mathcal{P}}$ corresponds to $\Delta;\phi, C \vdash_{\mathcal{T}}$.

□

References

[NOT06] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. of the ACM Press*, 53(6):937–977, 2006.

[Tin02] C. Tinelli. A DPLL-based calculus for ground satisfiability modulo theories. In G. Ianni and S. Flesca, editors, *Proc. of the 8th European Conf. on Logics in Artificial Intelligence*, volume 2424 of *LNAI*, pages 308–319. Springer-Verlag, 2002.